

Microprocessor and Computer Architecture

UE21CS251B

4th Semester, Academic Year 2022-23

Date: 08/02/23

Name: Nikhil Girish	SRN: PES2UG21CS334	Section: F
---------------------	--------------------	------------

Week# ____3____ Program Number: ____1__

Title of the Program

Generate Fibonacci Series and store them in an array.

I.ARM Assembly Code:

```
@ Fibonacci Sequence
.data
fib: .word 0,0,0,0,0,0,0,0,0,0

.text
MOV R3, #10
LDR R0, =fib
MOV R1, #0
MOV R2, #1
STR R1, [R0], #4
STR R2, [R0], #4
loop:
    ADD R4, R1, R2
    STR R4, [R0], #4
    MOV R1, R2
    MOV R2, R4
    SUB R3, R3, #1
    CMP R3, #2
    BNE loop
SWI 0x011
```

[illegible]

Microprocessor and Computer Architecture

UE21CS251B

4th Semester, Academic Year 2022-23

Date: 08/02/23

Name: Nikhil Girish	SRN: PES2UG21CS334	Section: F
---------------------	--------------------	------------

Week# 3 Program Number: 2

Title of the Program

Write an ALP to find smallest number in an array of n 32-bit numbers

I.ARM Assembly Code:

```
@Smallest number
.data
a: .word 9,53,1,7,33,56,48,93,90,51
b: .word -1

.text
LDR R0,=a
LDR R1,[R0],#4
LDR R4,=b
MOV R3,#1
l:
    LDR R2,[R0],#4
    CMP R1,R2
    MOVGT R1,R2
    ADD R3,R3,#1
    CMP R3,#9
    BNE l
    B exit
exit:
```

```
STR R1, [R4]
SWI 0x11
```

II. Output Screen Shots (One):

Registers View

General Purpose Floating Point

Hexadecimal

Unsigned Decimal

Signed Decimal

R0

:00001060

R1

:00000001

R2

:0000005a

R3

:00000009

R4

:00001064

R5

:00000000

R6

:00000000

R7

:00000000

R8

:00000000

R9

:00000000

R10 (s1)

:00000000

R11 (fp)

:00000000

R12 (ip)

:00000000

R13 (sp)

:00011400

R14 (lr)

:00000000

R15 (pc)

:00001030

CPSR Register

Negative (N)

:0

Zero (Z)

:1

Carry (C)

:1

Overflow (V)

:0

IRQ Disable

:1

FIQ Disable

:1

Thumb (T)

:0

CPU Mode

:System

Code View

P2.o

```

#Smallest number
.data
a: .word 9,53,1,7,33,56,48,93,90,51

:00000035
:00000001
:00000007
:00000021
00001064:FFFFFFF b: .word -1

.text
00001000:E59F002C LDR R0,=a
00001004:E4901004 LDR R1,[R0],#4
00001008:E59F4028 LDR R4,=b
0000100C:E3A03001 MOV R3,#1
1:
LDR R2,[R0],#4
CMP R1,R2
MOVGT R1,R2
ADD R3,R3,#1
CMP R3,#9
BNE 1
B exit
exit:
STR R1,[R4]
0000102C:E5841000
00001030:EF000011 SWI 0x11...

:00000000
:00000028

```

Output View WatchView MemoryView0

00001030

Word Size

8bit 16bit 32bit

00001030	EF000011	0000103C	00001064	00000009	00000035	00000001	00000007	00000021	00000038	00000030	0000005D	0000005A	00000033
00001064	00000001	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181
00001098	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181
000010cc	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181
00001100	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181
00001134	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181

0x600000df

Microprocessor and Computer Architecture

UE21CS251B

4th Semester, Academic Year 2022-23

Date: 08/02/23

Name: Nikhil Girish	SRN: PES2UG21CS334	Section: F
---------------------	--------------------	------------

Week# 3 Program Number: 3

Title of the Program

To perform Convolution using MUL instruction (Addition of multiplication of respective numbers of loc A and loc B)

I.ARM Assembly Code:

```
@Convolution using MUL
.data
a: .word 1,2,3,4,5,6,7,8,9
b: .word 10,20,30,40,50,60,70,80,90
c: .word 0

.text
LDR R0,=a
LDR R1,=b
LDR R2,=c
MOV R5,#0
MOV R6,#1
l:
    LDR R3,[R0],#4
    LDR R4,[R1],#4
    MUL R7,R3,R4
    ADD R5,R5,R7
    ADD R6,R6,#1
    CMP R6,#10
```

```

BNE 1
B end
end:
STR R5,[R2]
SWI 0x11

```

II. Output Screen Shot (One):

The screenshot displays a debugger interface with three main panes: RegistersView, CodeView, and MemoryView.

RegistersView: Shows the state of 16 registers (R0-R15) and CPSR. The CPSR register is expanded to show flags: Negative (N): 0, Zero (Z): 1, Carry (C): 1, Overflow (V): 0, IRQ Disable: 1, FIQ Disable: 1, Thumb (T): 0, and CPU Mode: System.

CodeView: Displays assembly code for a program titled "Convolution using MUL". The code includes data definitions for arrays 'a' and 'b', and a series of instructions including LDR, MUL, ADD, CMP, BNE, and B end.

MemoryView: Shows a memory dump starting at address 0000106C. The dump is organized into columns of 16-bit and 32-bit values, with the 32-bit column highlighted. The values in the 32-bit column are consistently 81818181.

Microprocessor and Computer Architecture

UE21CS251B

4th Semester, Academic Year 2022-23

Date: 08/02/23

Name: Nikhil Girish	SRN: PES2UG21CS334	Section: F
---------------------	--------------------	------------

Week# 3 Program Number: 4

Title of the Program

To perform Convolution using MLA instruction (Addition of multiplication of respective numbers of loc A and loc B).

I.ARM Assembly Code:

```
@Convolution using MLA
.data
a: .word 1,2,3,4,5,6,7,8,9
b: .word 10,20,30,40,50,60,70,80,90
c: .word 0

.text
LDR R0,=a
LDR R1,=b
LDR R2,=c
MOV R5,#0
MOV R6,#1
l:
    LDR R3,[R0],#4
    LDR R4,[R1],#4
    MLA R5,R3,R4,R5
    ADD R6,R6,#1
    CMP R6,#10
    BNE l
    B end
end:
```

```
STR R5,[R2]
SWI 0x11
```

II. Output Screen Shot (One):

The screenshot displays a debugger window with three main panes: RegistersView, CodeView, and MemoryView.

RegistersView: Shows the state of 16 registers (R0-R15) and CPSR. The values are as follows:

Register	Value
R0	00001068
R1	0000108c
R2	0000108c
R3	00000009
R4	0000005a
R5	00000b22
R6	0000000a
R7	00000000
R8	00000000
R9	00000000
R10(s1)	00000000
R11(fp)	00000000
R12(ip)	00000000
R13(sp)	00011400
R14(lr)	00000000
R15(pc)	00001034

CPSR Register:
Negative(N): 0
Zero(Z): 1
Carry(C): 1
Overflow(V): 0
IRQ Disable: 1
FIQ Disable: 1
Thumb(T): 0
CPU Mode: System

CodeView: Shows assembly code for a program titled "P4.o". The code is as follows:

```
@Convolution using MLA
.data
a: .word 1,2,3,4,5,6,7,8,9
b: .word 10,20,30,40,50,60,70,80,90
c: .word 0

.text
LDR R0,=a
LDR R1,=b
LDR R2,=c
MOV R5,#0
MOV R6,#1
L:
LDR R3,[R0],#4
LDR R4,[R1],#4
MLA R5,R3,R4,R5
ADD R6,R6,#1
CMP R6,#10
BNE 1
B end
end:
STR R5,[R2]
00001030: E5825000
```

MemoryView: Shows a memory dump starting at address 00001068. The data is as follows:

Address	Hex Value
00001068	0000000A
00001069	00000014
0000106A	0000001E
0000106B	00000028
0000106C	00000032
0000106D	0000003C
0000106E	00000046
0000106F	00000050
00001070	0000005A
00001071	000000B2
00001072	81818181
00001073	81818181
00001074	81818181
00001075	81818181
00001076	81818181
00001077	81818181
00001078	81818181
00001079	81818181
0000107A	81818181
0000107B	81818181
0000107C	81818181
0000107D	81818181
0000107E	81818181
0000107F	81818181
00001080	81818181
00001081	81818181
00001082	81818181
00001083	81818181
00001084	81818181
00001085	81818181
00001086	81818181
00001087	81818181
00001088	81818181
00001089	81818181
0000108A	81818181
0000108B	81818181
0000108C	81818181
0000108D	81818181
0000108E	81818181
0000108F	81818181
00001090	81818181
00001091	81818181
00001092	81818181
00001093	81818181
00001094	81818181
00001095	81818181
00001096	81818181
00001097	81818181
00001098	81818181
00001099	81818181
0000109A	81818181
0000109B	81818181
0000109C	81818181
0000109D	81818181
0000109E	81818181
0000109F	81818181
000010A0	81818181
000010A1	81818181
000010A2	81818181
000010A3	81818181
000010A4	81818181
000010A5	81818181
000010A6	81818181
000010A7	81818181
000010A8	81818181
000010A9	81818181
000010AA	81818181
000010AB	81818181
000010AC	81818181
000010AD	81818181
000010AE	81818181
000010AF	81818181
000010B0	81818181
000010B1	81818181
000010B2	81818181
000010B3	81818181
000010B4	81818181
000010B5	81818181
000010B6	81818181
000010B7	81818181
000010B8	81818181
000010B9	81818181
000010BA	81818181
000010BB	81818181
000010BC	81818181
000010BD	81818181
000010BE	81818181
000010BF	81818181
000010C0	81818181
000010C1	81818181
000010C2	81818181
000010C3	81818181
000010C4	81818181
000010C5	81818181
000010C6	81818181
000010C7	81818181
000010C8	81818181
000010C9	81818181
000010CA	81818181
000010CB	81818181
000010CC	81818181
000010CD	81818181
000010CE	81818181
000010CF	81818181
000010D0	81818181
000010D1	81818181
000010D2	81818181
000010D3	81818181
000010D4	81818181
000010D5	81818181
000010D6	81818181
000010D7	81818181
000010D8	81818181
000010D9	81818181
000010DA	81818181
000010DB	81818181
000010DC	81818181
000010DD	81818181
000010DE	81818181
000010DF	81818181
000010E0	81818181
000010E1	81818181
000010E2	81818181
000010E3	81818181
000010E4	81818181
000010E5	81818181
000010E6	81818181
000010E7	81818181
000010E8	81818181
000010E9	81818181
000010EA	81818181
000010EB	81818181
000010EC	81818181
000010ED	81818181
000010EE	81818181
000010EF	81818181
000010F0	81818181
000010F1	81818181
000010F2	81818181
000010F3	81818181
000010F4	81818181
000010F5	81818181
000010F6	81818181
000010F7	81818181
000010F8	81818181
000010F9	81818181
000010FA	81818181
000010FB	81818181
000010FC	81818181
000010FD	81818181
000010FE	81818181
000010FF	81818181
00001100	81818181
00001101	81818181
00001102	81818181
00001103	81818181
00001104	81818181
00001105	81818181
00001106	81818181
00001107	81818181
00001108	81818181
00001109	81818181
0000110A	81818181
0000110B	81818181
0000110C	81818181
0000110D	81818181
0000110E	81818181
0000110F	81818181
00001110	81818181
00001111	81818181
00001112	81818181
00001113	81818181
00001114	81818181
00001115	81818181
00001116	81818181
00001117	81818181
00001118	81818181
00001119	81818181
0000111A	81818181
0000111B	81818181
0000111C	81818181
0000111D	81818181
0000111E	81818181
0000111F	81818181
00001120	81818181
00001121	81818181
00001122	81818181
00001123	81818181
00001124	81818181
00001125	81818181
00001126	81818181
00001127	81818181
00001128	81818181
00001129	81818181
0000112A	81818181
0000112B	81818181
0000112C	81818181
0000112D	81818181
0000112E	81818181
0000112F	81818181
00001130	81818181
00001131	81818181
00001132	81818181
00001133	81818181
00001134	81818181
00001135	81818181
00001136	81818181
00001137	81818181
00001138	81818181
00001139	81818181
0000113A	81818181
0000113B	81818181
0000113C	81818181
0000113D	81818181
0000113E	81818181
0000113F	81818181
00001140	81818181
00001141	81818181
00001142	81818181
00001143	81818181
00001144	81818181
00001145	81818181
00001146	81818181
00001147	81818181
00001148	81818181
00001149	81818181
0000114A	81818181
0000114B	81818181
0000114C	81818181
0000114D	81818181
0000114E	81818181
0000114F	81818181
00001150	81818181
00001151	81818181
00001152	81818181
00001153	81818181
00001154	81818181
00001155	81818181
00001156	81818181
00001157	81818181
00001158	81818181
00001159	81818181
0000115A	81818181
0000115B	81818181
0000115C	81818181
0000115D	81818181
0000115E	81818181
0000115F	81818181
00001160	81818181
00001161	81818181
00001162	81818181
00001163	81818181
00001164	81818181
00001165	81818181
00001166	81818181
00001167	81818181
00001168	81818181
00001169	81818181
0000116A	81818181
0000116B	81818181
0000116C	81818181
0000116D	81818181
0000116E	81818181
0000116F	81818181
00001170	81818181
00001171	81818181
00001172	81818181
00001173	81818181
00001174	81818181
00001175	81818181
00001176	81818181
00001177	81818181
00001178	81818181
00001179	81818181
0000117A	81818181
0000117B	81818181
0000117C	81818181
0000117D	81818181
0000117E	81818181
0000117F	81818181
00001180	81818181
00001181	81818181
00001182	81818181
00001183	81818181
00001184	81818181
00001185	81818181
00001186	81818181
00001187	81818181
00001188	81818181
00001189	81818181
0000118A	81818181
0000118B	81818181
0000118C	81818181
0000118D	81818181
0000118E	81818181
0000118F	81818181
00001190	81818181
00001191	81818181
00001192	81818181
00001193	81818181
00001194	81818181
00001195	81818181
00001196	81818181
00001197	81818181
00001198	81818181
00001199	81818181
0000119A	81818181
0000119B	81818181
0000119C	81818181
0000119D	81818181
0000119E	81818181
0000119F	81818181
000011A0	81818181
000011A1	81818181
000011A2	81818181
000011A3	81818181
000011A4	81818181
000011A5	81818181
000011A6	81818181
000011A7	81818181
000011A8	81818181
000011A9	81818181
000011AA	81818181
000011AB	81818181
000011AC	81818181
000011AD	81818181
000011AE	81818181
000011AF	81818181
000011B0	81818181
000011B1	81818181
000011B2	81818181
000011B3	81818181
000011B4	81818181
000011B5	81818181
000011B6	81818181
000011B7	81818181
000011B8	81818181
000011B9	81818181
000011BA	81818181
000011BB	81818181
000011BC	81818181
000011BD	81818181
000011BE	81818181
000011BF	81818181
000011C0	81818181
000011C1	81818181
000011C2	81818181
000011C3	81818181
000011C4	81818181
000011C5	81818181
000011C6	81818181
000011C7	81818181
000011C8	81818181
000011C9	81818181
000011CA	81818181
000011CB	81818181
000011CC	81818181
000011CD	81818181
000011CE	81818181
000011CF	81818181
000011D0	81818181
000011D1	81818181
000011D2	81818181
000011D3	81818181
000011D4	81818181
000011D5	81818181
000011D6	81818181
000011D7	81818181
000011D8	81818181
000011D9	81818181
000011DA	81818181
000011DB	81818181
000011DC	81818181
000011DD	81818181
000011DE	81818181
000011DF	81818181
000011E0	81818181
000011E1	81818181
000011E2	81818181
000011E3	81818181
000011E4	81818181
000011E5	81818181
000011E6	81

Microprocessor and Computer Architecture

UE21CS251B

4th Semester, Academic Year 2022-23

Date: 08/02/23

Name: Nikhil Girish	SRN: PES2UG21CS334	Section: F
---------------------	--------------------	------------

Week# 3 Program Number: 5

Title of the Program

Write an ALP to find mul(add(a,b),c)

I.ARM Assembly Code:

```
@mul<add(a,b),c>
.data
a: .word 0
stk: .word 0

.text
LDR R0,=a
MOV R1,#10
MOV R2,#20
MOV R3,#30
BL mula
STR R6,[R0]
B end

mula:
LDR R4,=stk
STR LR,[R4]
BL add
MUL R6,R5,R3
LDR LR,[R4]
MOV PC,LR
```

```

add:
    ADD R5,R2,R1
    MOV PC,LR

end:
    SWI 0x011

```

II. Output Screen Shot (One):

The screenshot displays a debugger window with the following components:

- RegistersView:** Shows the state of 16 registers (R0-R15). R0 is highlighted in red. The CPSR register shows flags: Negative (N): 0, Zero (Z): 0, Carry (C): 0, Overflow (V): 0, IRQ Disable: 1, FIQ Disable: 1, Thumb (T): 0, CPU Mode: System.
- CodeView:** Displays assembly code for a file named 'P5.o'. The code includes labels 'add:', 'end:', and 'SWI 0x011...'. It also shows data and text sections with various instructions like LDR, MOV, BL, STR, and MVA.
- MemoryView:** Shows a memory dump starting at address 0x000000dF. The dump displays hexadecimal values in columns, with the first column showing addresses from 00001048 to 0000114C.

Microprocessor and Computer Architecture

UE21CS251B

4th Semester, Academic Year 2022-23

Date: 08/02/23

Name: Nikhil Girish	SRN: PES2UG21CS334	Section: F
---------------------	--------------------	------------

Week# 3 Program Number: 6

Title of the Program

Write an ALP to find factorial using subroutine

I.ARM Assembly Code:

```
@Factorial
.data
a: .word 0

.text
LDR R0,=a
MOV R1,#10
BL fact
STR R2,[R0]
B end

fact:
    MOV R2,#1

l:
    MUL R2,R2,R1
    SUB R1,R1,#1
    CMP R1,#0
    BGT l
    MOV PC,LR

end:
```

SWI 0x011

II. Output Screen Shot (One):

The screenshot displays a debugger interface with the following components:

- RegistersView:** Shows the state of 16 registers (R0-R15) and the CPSR register. R0 is highlighted with a red background and contains the value 00001034. The CPSR register shows flags: Negative(N):0, Zero(Z):1, Carry(C):1, Overflow(O):0, IRQ Disable:1, FIQ Disable:1, Thumb(T):0, and CPU Mode: System.
- CodeView:** Displays assembly code for a program named P6.o. The code includes a factorial function and a main function. The instruction at address 0000102C is highlighted in blue: `SWI 0x011...`.
- MemoryView:** Shows the memory dump starting at address 00001034. The first few bytes are 00375F00, followed by a series of 01s.

The assembly code in CodeView is as follows:

```
#factorial
.data
a: .word 0

.text
00001000:E59F0028 LDR R0,=a
00001004:E3A0100A MOV R1,#10
00001008:EB000001 BL fact
0000100C:E5802000 STR R2,[R0]
00001010:EAD00003 B end

fact:
00001014:E3A02001 MOV R2,#1

l:
00001018:E0020192 MUL R2,R2,R1
0000101C:E2411001 SUB R1,R1,#1
00001020:E3510000 CMP R1,#0
00001024:CAFFFFFFB BGT l
00001028:E1A0F008 MOV PC,LR

end:
0000102C:EF000011 SWI 0x011...
:00000000
```

Microprocessor and Computer Architecture

UE21CS251B

4th Semester, Academic Year 2022-23

Date: 08/02/23

Name: Nikhil Girish	SRN: PES2UG21CS334	Section: F
---------------------	--------------------	------------

Week# 3 Program Number: 7

Title of the Program

Write an ALP to perform multiplication using shift method (without using MUL)

I.ARM Assembly Code:

```
@MUL using shift
.text
MOV r0,#4
MOV r2,r0,LSL #4
SUB r2,r2,r0,LSL #3
SWI 0x011
```

II. Output Screen Shot (One):

The screenshot displays a debugger interface with three main panes: RegistersView, CodeView, and MemoryView.

RegistersView: Shows the state of various registers. R0 is highlighted in red with a value of 00000004. R15 (pc) is also highlighted in red with a value of 0000100c. The CPSR Register shows various flags: Negative (N): 0, Zero (Z): 0, Carry (C): 0, Overflow (V): 0, IRQ Disable: 1, FIQ Disable: 1, Thumb (T): 0, and CPU Mode: System.

CodeView: Displays assembly code for file P7.o. The code includes instructions like MOV r0, #4, MOV r2, r0, LSL #4, SUB r2, r2, r0, LSL #3, and SWI 0x011...

MemoryView: Shows a memory dump starting at address 0000100c. The data is displayed in hexadecimal and binary (01010101) format. The word size is set to 32bit.

Disclaimer:

- The programs and output submitted is duly written, verified and executed by me.
- I have not copied from any of my peers nor from the external resource such as internet.
- If found plagiarized, I will abide with the disciplinary action of the University.

Signature: Nikhil Girish

Name: Nikhil

SRN: PES2UG21CS334

Section: 4F

Date: 08/02/23